AMMS Practical: Estimating Population Structure from Allele Frequency Data

Izzy Routledge, Bob Verity, Nick Brazeau

August 05, 2022

Contents

Dependencies for Practical

Please copy and paste the below code chunk in it's entirety to your console to download R package libraries needed for this practical. If you are having trouble installing any of the R packages, please ask an instructor for a pre-loaded flash drive.

```r
deps <- c("tidyverse", "genescaperlite", "MIPanalyzer","rmaverick")
deps <- !sapply(deps, function(x){x %in% installed.packages()[,1]})
if (any(deps)) {
  if (deps["tidyverse"]) {
    install.packages("tidyverse")
  }
  if (deps["genescaper"]) {
    devtools::install_github("nickbrazeau/genescaperlite", ref = "v0.1.0")
  }
  if (deps["MIPanalyzer"]) {
    devtools::install_github("mrc-ide/MIPanalyzer", ref = "v1.0.0")
  }
  if (deps["rmaverick"]) {
    devtools::install_github("bobverity/rmaverick", ref = "v1.1.0")
  }
}
```

Now load all of those libraries into this session using the code chunk below. Please copy and paste it in its entirety.

```r
library(tidyverse)
library(genescaperlite)
library(MIPanalyzer)
library(rmaverick)
```

Finally, source the additional functions that are needed for this practical by copy-pasting this function:

```r
source("source_functions/pop_structure_utils.R")
```

Intro to population structure and allele frequency metrics

One of the most fundamental observations that we can make from genetic data is how the frequencies of different alleles vary over space and time.

For loci where mutations confer drug resistance, allele frequencies tell us something clinically important: if an allele is at high frequencies then we might consider performing a therapeutic efficacy study (TES) and eventually switching first-line drugs.

However, even for loci that have no direct impact on phenotype, and therefore are selectively neutral, allele frequencies can give us important information about what is going on in the parasite population.

These loci will be influenced by factors such as prevalence, human and mosquito population size and migration rates between different sub-populations. If we can correctly interpret these signals then we can use neutral allele frequencies to provide information about population structure and connectivity that can be relevant for control purposes.

Definition: *A locus is a fixed position on a chromosome where a particular genetic marker is located.*

Definition: *An allele is a particular variant of a gene or genetic locus.*

Definition: *A neutral allele is one that does not positively or negatively affect an organism's fitness.*

Overview of Data

We will work with both simulated and real data in this practical. The simulated data will allow us to explore how allele frequencies change over time as a function of different underlying parameters - something that we can almost never do from messy real-world data. This will allow us to build up an intuition about what we would expect to see in real data. In the second half of the practical, we will use real-world datasetes including a large molecular inversion probe (MIP) dataset taken from DRC and surrounding countries, previously analysed by Verity et al. 2020, and a smaller genetic barcode dataset taken from Senegal, previously analysed by Bei et al., 2018.

Practical Goals

By the end of this practical, you should be able to:

- Describe how population size, mutation and migration rates affect the patterns that we expect to see in neutral allele frequency data
- Estimate metrics such as F_{ST} and explain what they can tell us about population differentiation
- Use ordination techniques like PCA to explore patterns of population structure
- Use model-based techniques like *rmaverick* to detect population structure probabilistically
- Appreciate how allele frequency methods can identify connectivity over large spatial scales and long temporal scales
- Explain how allele frequencies can give useful information, and also describe their limitations for malaria control purposes

---

Population genetics through simulation

In this section, we will simulate various populations with different frequencies of alleles be using the R function: `sim_freqs()`. Note, this function is not present in base R, but was loaded from `source_functions/pop_structure_utils.R` which in turn references the `genescaper` package (in development). This function simulates data from a Wright-Fisher model, which is a general population genetic model and not specific to malaria. It therefore fails to capture some dynamics of malaria, such as superinfection, but it has the advantage of being very simple and easy to interpret.

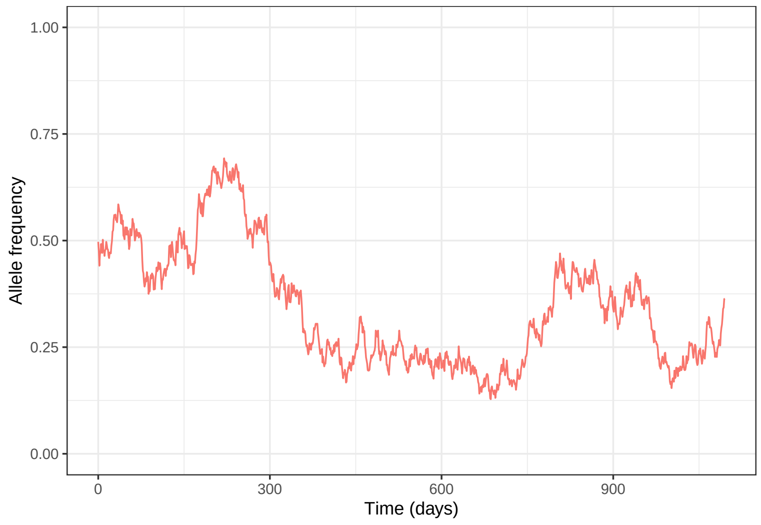
Single subpopulation

Let's first consider a single "deme", or subpopulation. We can think of this as a small population of infected hosts, with parasites being freely transmitted from any individual to any other (i.e. no human or mosquito population structure).

Allele frequency changes in a single deme

First we will simulate the allele frequencies of a population of 1000 infected individuals:

```r
sim1 <- sim_freqs(N = 1000, mut_rate = 0)
```



Q1. Try running this function lots of times. What do you notice about the allele frequency over time? What happens if they reach exactly 0 or 1, and why?

Click For Answer

A1. Allele frequencies fluctuate up and down in a random fashion over time. This is a result of our assumption of finite population size and random mating, which means some infections will be more "successful" than others, passing on more secondard infections. Results are different each time we run the function due to the random nature of the process. Sometimes allele frequencies may reach exactly 0 or 1 ("loss" or "fixation"), in which case they then *stay at this level*. This is because we have assumed no mutation rate above, meaning once a population is fixed for a single allele there is no way of new variation entering back in.

Q2. Now repeat with a smaller population size (which represents the number of *infected* hosts in our case); for example, `N=500`, `N=100`, and `N=50`. What do you notice about the strength of drift?

Click For Answer

A2. The strength of genetic drift (i.e. the magnitude of fluctuations in allele frequencies) is greatest when population size is small. We have a greater chance of reaching fixation or loss more quickly when population size is small. From a malaria point of view, this means we are more likely to see a single allele dominate a population by pure chance when the number of infected hosts is small (i.e. low prevalence or small population size).

Definition: *Genetic drift is the process of random fluctiations in allele frequencies in a finite population.*

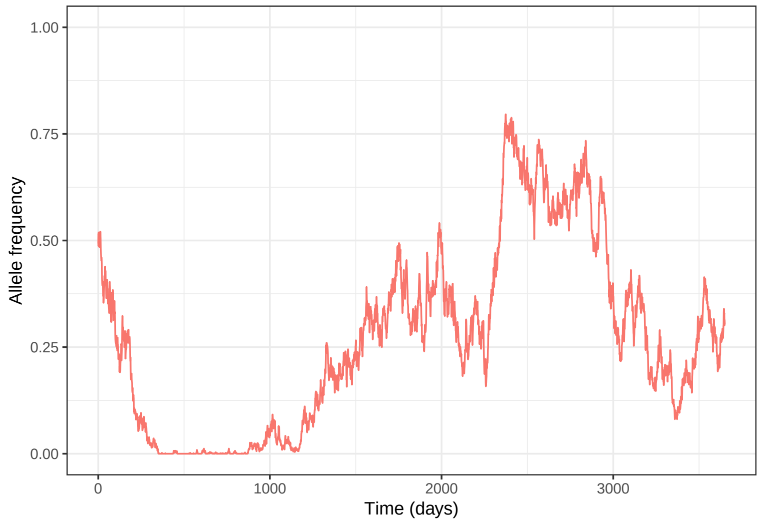
So far we have assumed no mutation, meaning the only force affecting allle frequencies is drift. Next, we will explore how our simulation results change when we add mutation. The mutation rates we will consider here are several orders of magnitude larger than typical SNP mutation rates in *P. falciparum*, which are around `1e-9`. This is for convenience only - we could get the same results with a smaller mutation rate, but we would need a larger population size to see the effects which would be slow to simulate.

Q3. Repeat the simulation above, this time running for 10 years. Start by assuming a population size of 1000 infected individuals and a mutation rate of `1e-4`. Do frequencies still always end up fixed/lost? What happens if you increase the mutation rate? What happens if you decrease the population size?

Click For Answer

A3. Alleles tend to reach 0 and 1 over such a long period of time, but they then return to intermediate frequencies. This is due to mutation reintroducing a small amount of variation, which drift can then take back up to high frequencies. Higher mutation rates mean the allele frequencies tend to stay at intermediate values for longer. Smaller population sizes mean drift is stronger (as above), so allele frequencies tend to remain close to 0 or 1.

```
sim_freqs(N = 1000, t_out = seq(0, 365 * 10), mut_rate = 1e-4)
```



The allele frequency distribution

We will now simulate a large number of loci, but output will just consider a single timepoint: 50 years from the start of our simulation.

```
# simulate allele frequencies and look at the data
sim1 <- sim_freqs(N = 1e3, t_out = 365 * 50, mut_rate = 1e-4, loci = 1e3, plot_on = FALSE)
head(sim1$data)
```

```
##    time deme locus  freq
## 1 18250    1     1 0.999
## 2 18250    1     2 0.116
## 3 18250    1     3 0.999
## 4 18250    1     4 0.552
## 5 18250    1     5 0.011
## 6 18250    1     6 0.035
```

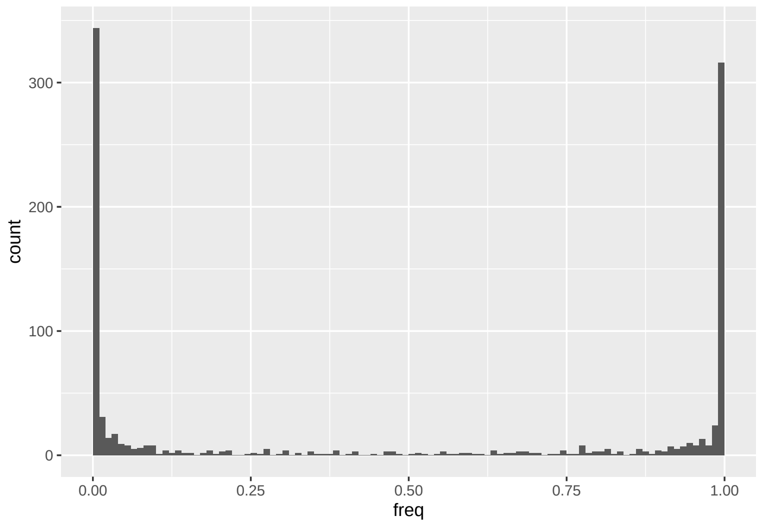
We can see that our different loci reach different allele frequencies due to chance. It can be useful to look at the **allele frequency distribution**, sometimes also called the **allele frequency spectrum**, to see how often we see each frequency.

Q4. Using the simulated data located in the `sim1$data` dataframe, make a histogram of allele frequencies. Try using ggplot with the `geom_histogram()` function. You may need to modify the `breaks` argument of this function to see the frequency distribution clearly.

Click For Answer

A4. Example code is shown below:

```
ggplot(sim1$data) +
  geom_histogram(aes(x = freq), breaks = seq(0, 1, 0.01))
```

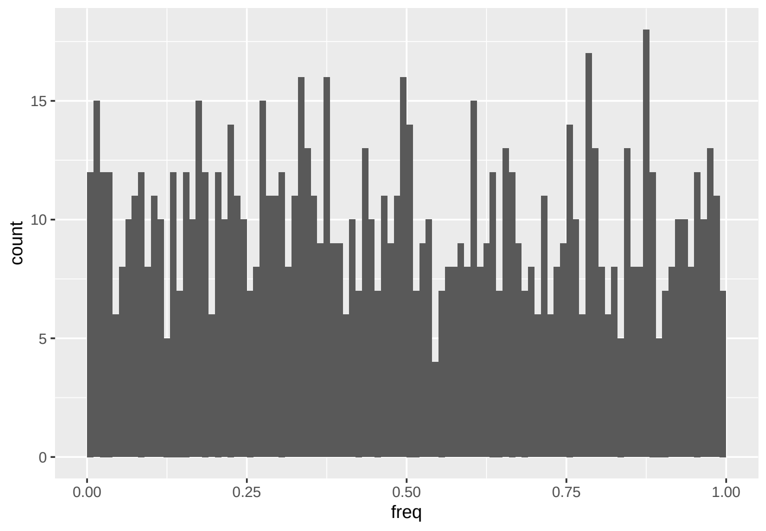


Q5. What happens to the allele frequency distribution if you increase the mutation rate in your simulation? What happens if you increase the population size?

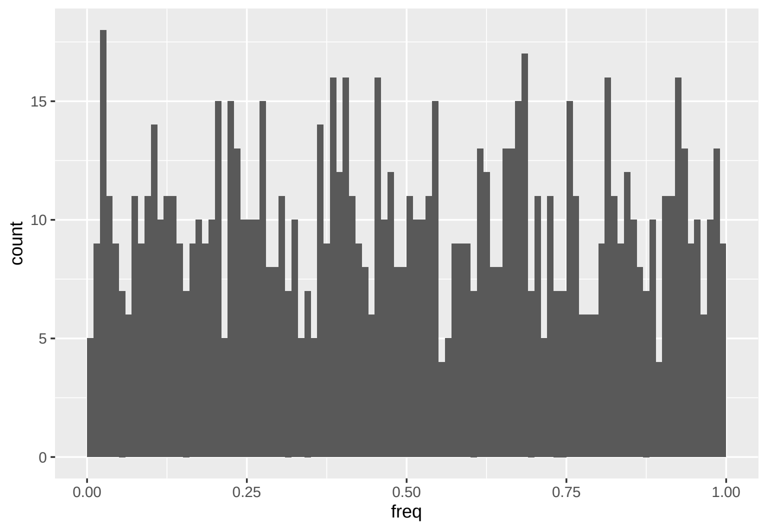
Click For Answer

A5. The example code below shows simulations with higher mutation rate or larger population size than previously. In both cases we see that the allele frequency distribution is more concentrated at central values, rather than being pushed up against the walls at 0 and 1. This is becaues mutation is strong *relative to drift*, meaning intermediate allele frequencies can be sustained.

```
# simulate and plot large mutation rate
sim2 <- sim_freqs(N = 1e3, t_out = 365 * 50, mut_rate = 1e-3, loci = 1e3, plot_on = FALSE)
ggplot(sim2$data) +
  geom_histogram(aes(x = freq), breaks = seq(0, 1, 0.01))
```



```r
# simulate and plot large population size
sim3 <- sim_freqs(N = 1e4, t_out = 365 * 50, mut_rate = 1e-4, loci = 1e3, plot_on = FALSE)
ggplot(sim3$data) +
  geom_histogram(aes(x = freq), breaks = seq(0, 1, 0.01))
```



Real-world SNP allele frequency distributions tend to look more like our first simulation, with most values close to 0 or 1. This is because the effective population size in *P. falciparum* is very large, and the mutation rate is also fairly small. Remember that this distribution is what we would expect for *neutral* alleles only - things like balancing selection can create more complicated patterns, tending to maintain frequencies at intermediate values for long periods.

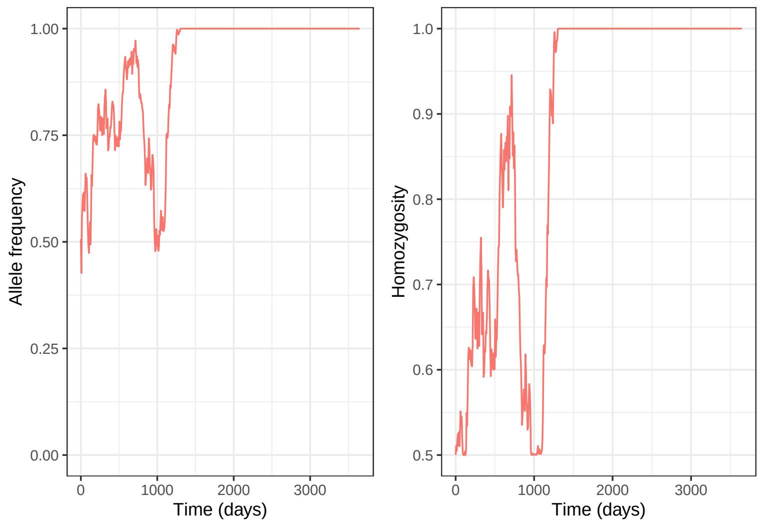
The buildup of homozygosity

Now we will analyze how homozygosity changes over time as a consequence of genetic drift.

Definition: *Homozygosity is the existence of identical allleles at a given locus. It can also be used to describe the frequency or probability of seeing identical alleles.*

Q6. Run the code below multiple times to repeat the simulation of drift with no mutation, but now also plotting homozygosity. What do you notice about how homozygosity changes over time?

```
sim_freqs(N = 1000, t_out = seq(0, 365 * 10, 7), plot_homo = TRUE)
```



Click For Answer

A6. Homozygosity tends to increase to 1 over time. This is true irrespective of the direction of allele frequency drift (i.e. towards 0 or towards 1). Homozygosity reaches its final value of 1 at the time when allele frequencies fix at 0 or 1.

Q7. Run the above code but now consider what happens when you add in some mutation? Try mutation rates of `1e-3` and `1e-4`. What happens to the equilibrium level of homozygosity?

Click For Answer

A7. Allele frequencies tend to pulled towards intermediate values, meaning homozygosity is pulled away from 1. Even if homozygosity eventually reaches 1, mutation can mean that variation is reintroduced so homozygosity drops again. The equilibrium level of homozygosity is therefore not exactly 1 when there is some mutation present.

```
# explore homozygosity trends when there is some mutation
sim_freqs(N = 1000, t_out = seq(0, 365 * 10, 7), mut_rate = 1e-3, plot_homo = TRUE)
sim_freqs(N = 1000, t_out = seq(0, 365 * 10, 7), mut_rate = 1e-4, plot_homo = TRUE)
```

Multiple subpopulations

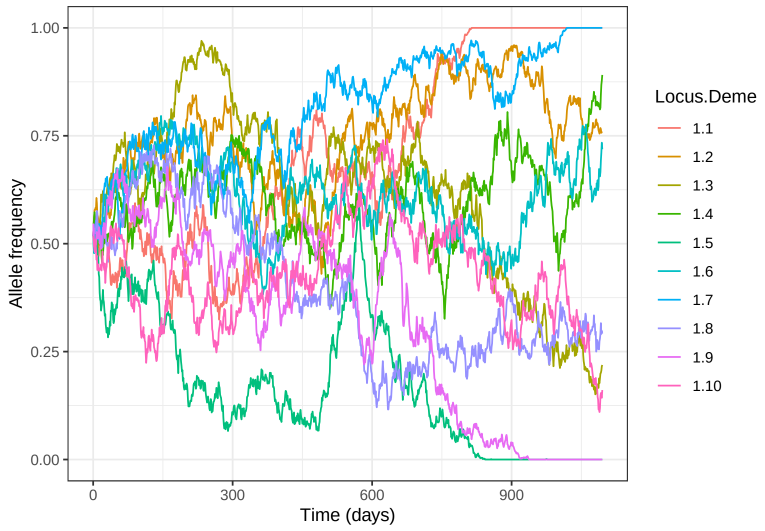
We have seen how allele frequencies change over time due to mutation and drift in a single subpopulation. In this section, we will explore how these patterns vary between subpopulations, and how migration influences these patterns.

Perfectly isolated subpopulations

First, we will explore a scenario where we have 10 perfectly independent demes. This could represent ten villages where there is no movement of people (or parasites) between those villages.

Run the following code, which simulates drift in 10 demes:

```
sim1 <- sim_freqs(N = 1000, demes = 10, mig_rate = 0)
```



You should see that allele frequencies tend to diverge between demes. This is the same behavior as our single subpopulation simulations above, but now replicated 10 times. Alleles frequencies have an equal chance of going up or down on any given day, meaning some demes will go up and some will go down and so overall we see divergence.

Q8. What would you expect to see with a larger population size? What about if you introduce mutation?

Click For Answer

A8. With a larger population size drift is weaker, meaning populations diverge but it takes longer. With some mutation we find that allele frequencies are pulled back towards intermediate values, meaning there is a limit to divergence (we don't always end up with alleles either fixed or lost in every deme).

```
sim1 <- sim_freqs(N = 1e4, demes = 10, mig_rate = 0)
sim1 <- sim_freqs(N = 1e3, demes = 10, mig_rate = 0, mut_rate = 1e-4)
```

Calculating Population Differentiation Next, we will explore metrics used to measure population differentiation. We will focus on two different metrics: the *fixation index*, or F_{ST} and *Jost's D*.

High values of F_{ST} and Jost's D both suggest a greater genetic differentiation between subpopulations. However both metrics have different approaches and different definitions of what exactly we mean by genetic "differentiation".

Fst We will first look at F_{ST}, which measures how close our populations are to fixation. As our populations become more differentiated, in the sense that they build up local ancestry, they will also approach fixation (assuming zero mutation). Therefore, it is common to interpret F_{ST} as a measure of differentiation as well as a measure of progress towards fixation.

Below, we have provided a simple version of F_{ST} that applies to biallelic loci only and can be calculated manually:

$$F_{ST} = \frac{\sigma^2_{\text{subpop}}}{\sigma^2_{\text{total}}} = \frac{\sigma^2_{\text{subpop}}}{\bar{p}(1-\bar{p})}$$

This equation says that F_{ST} is a measure of the variance of allele frequencies among the subpopulations divided by the total variation in the allele frequency in the population.

```
# simulate allele frequencies as above
sim1 <- sim_freqs(N = 1e3, demes = 10, t_out = seq(0, 365*10, 7), mig_rate = 0, plot_on = FALSE)

# calculate Fst at each timepoint
```

```r
sim_fst <- sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(p_mean = mean(freq),
            p_var = var(freq),
            Fst = p_var / (p_mean * (1 - p_mean)))

# look at first few results
head(sim_fst)
```

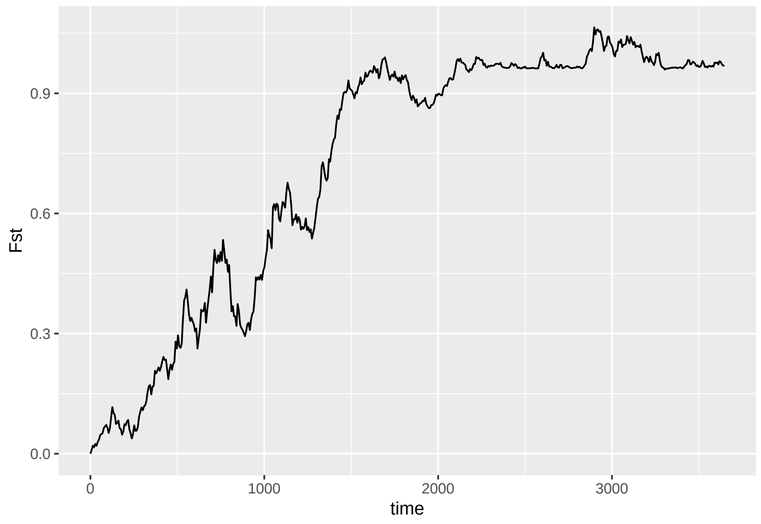
```
## # A tibble: 6 x 4
##    time p_mean   p_var     Fst
##   <dbl>  <dbl>   <dbl>   <dbl>
## 1     0  0.454 0       0
## 2     7  0.452 0.00245 0.00989
## 3    14  0.431 0.00497 0.0203
## 4    21  0.454 0.00404 0.0163
## 5    28  0.466 0.00608 0.0244
## 6    35  0.452 0.00487 0.0197
```

Q9. Can you use the ggplot2 function `geom_line()` to plot F_{ST} over time (x-axis: time, y-axis: Fst). What do you notice about differentiation?

Click For Answer

A9. Differentiation (as measured by Fst) increases over time. This is due to allele frequencies diverging, causing the variance between demes to make up an increasing proportion of the total variance.

```r
ggplot(sim_fst) +
  geom_line(aes(x = time, y = Fst)) +
  expand_limits(y = c(0, 1))
```

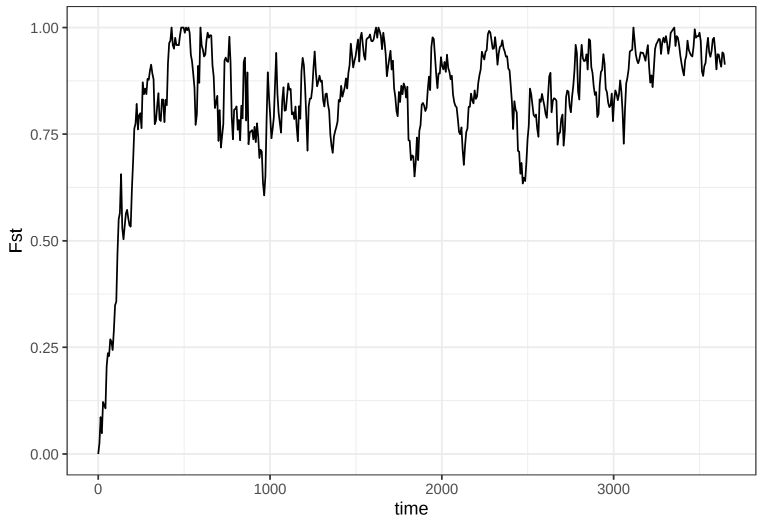


Q10. Now repeat the process above (simulation and plotting) with a population size of `N = 1e2` and a mutation rate of `mut_rate = 1e-3`. What changes do you notice?

Click For Answer

A10. F_{ST} increases over time, but does not level out at 1. Instead it bounces around at an equilibrium value less than 1.

```r
# simulate with a high mutation rate
sim1 <- sim_freqs(N = 1e2, demes = 10, mig_rate = 0, mut_rate = 1e-3,
                  t_out = seq(0, 365*10, 7), plot_on = FALSE)

# calculate Fst and plot results
sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(p_mean = mean(freq),
            p_var = popvar(freq),
            Fst = p_var / (p_mean * (1 - p_mean))) %>%
  ggplot() + theme_bw() +
  geom_line(aes(x = time, y = Fst)) +
  expand_limits(y = c(0, 1))
```



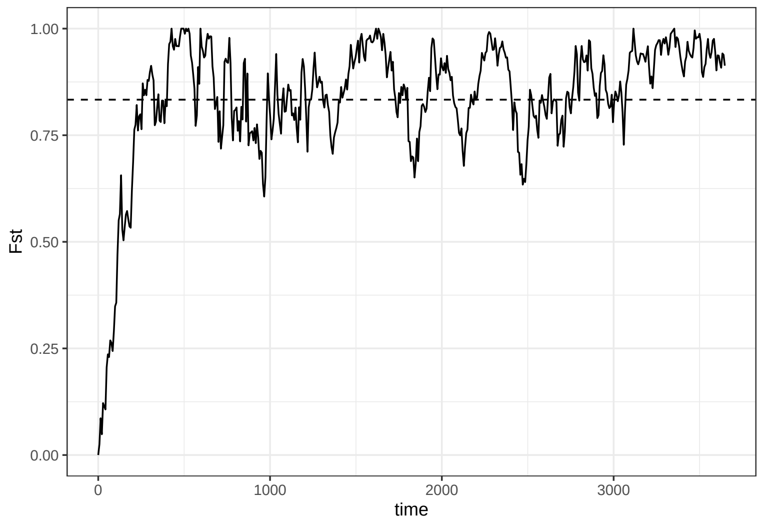
The equilibrium level of F_{ST} between drift and mutation is known to be $\frac{1}{(1+2N\mu)}$ for haploid organisms, where N is the population size, and μ is the mutation rate.

Q11. Add this level in the plot above using the `geom_hline()` function. How does this theoretical result compare with what you see in the plot?

Click For Answer

A11. The theoretical result matches what we observe in the plot very closely.

```r
# plot results as above, adding line for theoretical expectation
sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(p_mean = mean(freq),
            p_var = popvar(freq),
            Fst = p_var / (p_mean * (1 - p_mean))) %>%
  ggplot() + theme_bw() +
  geom_line(aes(x = time, y = Fst)) +
  expand_limits(y = c(0, 1)) +
  geom_hline(yintercept = 1 / (1 + 2*1e2*1e-3), linetype = "dashed")
```



The fact that F_{ST} depends on the mutation rate can lead to problems. If one marker has a high mutation rate (e.g. microsatellites) then we would expect to see low values of F_{ST}, even if the populations have been evolving separately for a long time, and in this sense are well differentiated. For this reason, we may prefer to look at other measures.

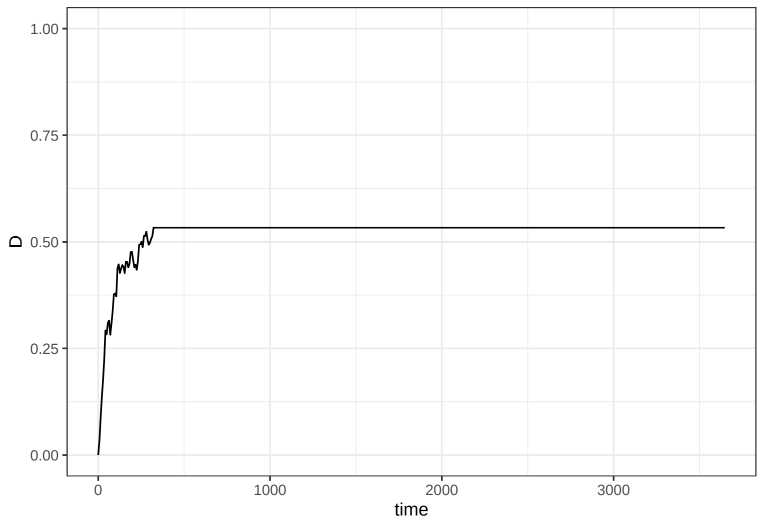
Jost's D Jost's D is an alternative way of looking at differentiation that does not suffer from the same issues with mutation. We can calculate Jost's D, using the follow equation:

$$D = \frac{H_T - H_S}{1 - H_S} \frac{k}{k-1}$$

Jost's D measures the extent of allele sharing between demes. When all alleles are "private" to a single deme, meaning the same allele is never seen in more than one deme, then Jost's D equals 1. This is a measure of "differentiation" in terms of partitioning the observed genetic diversity, and so is a completely different interpretation to F_{ST}.

We can use the code chunk below to explore this statistic:

```
# simulate with a high mutation rate
sim1 <- sim_freqs(N = 1e2, demes = 10, mig_rate = 0, mut_rate = 0,
                  t_out = seq(0, 365*10, 7), plot_on = FALSE)

# calculate Jost's D and plot results
sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(Hs = 1 - mean(freq^2 + (1 - freq)^2),
            Ht = 1 - (mean(freq)^2 + mean(1 - freq)^2),
            k = max(deme),
            D = (Ht - Hs) / (1 - Hs) * k / (k - 1)) %>%
  ggplot() + theme_bw() +
  geom_line(aes(x = time, y = D)) +
  expand_limits(y = c(0, 1.0))
```



Jost's D does not suffer from the same problem as F_{ST}, but it suffers from a different problem - it assumes infinite alleles mutation. When we have a biallelic SNPs, Jost's D will tend to level out at a value that is less than 1, and is also hard to predict. It would be a much more appropriate statistic for something like a microsatellite, where the inifinite alleles mutation is more reasonable.

In conclusion, both F_{ST} and D have their limitations. F_{ST} is good when mutation rates are low, for example biallelic SNPs. Jost's D is good for highly diverse and fast-mutating sites, such as microsatellites.

Connected subpopulations

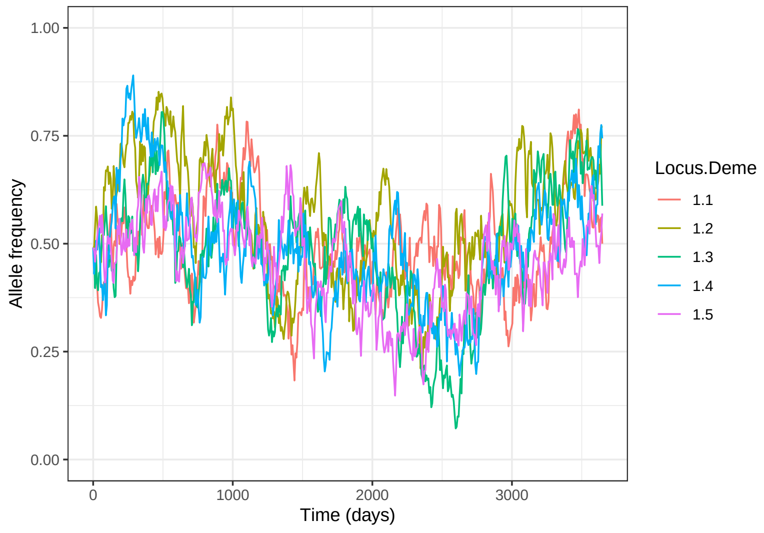
In this section, we will explore what happens when we connect our demes by migration. In other words, we are relaxing our assumption of independent populations from above.

Q12. Run the simulator now with 1000 individuals, 5 demes, a migration rate of 1 in 100 and a mutation rate of 1 in 10,000. What do you observe about the allele frequencies? Try experimenting with larger and smaller migration rates. How does this change the results?

Click For Answer

A12. With migration, allele frequencies still fluctuate but tend to move together. The higher the migration rate, the more closely correlated the results. You might also notive that fluctuations are smaller (drift is weaker) when migration is high, because our "effective" population size becomes larger.

```r
# simulation and plot with migration
sim_freqs(N = 1000, t_out = seq(0, 365*10, 7), demes = 5, mut_rate = 1e-4, mig_rate = 1e-2)
```



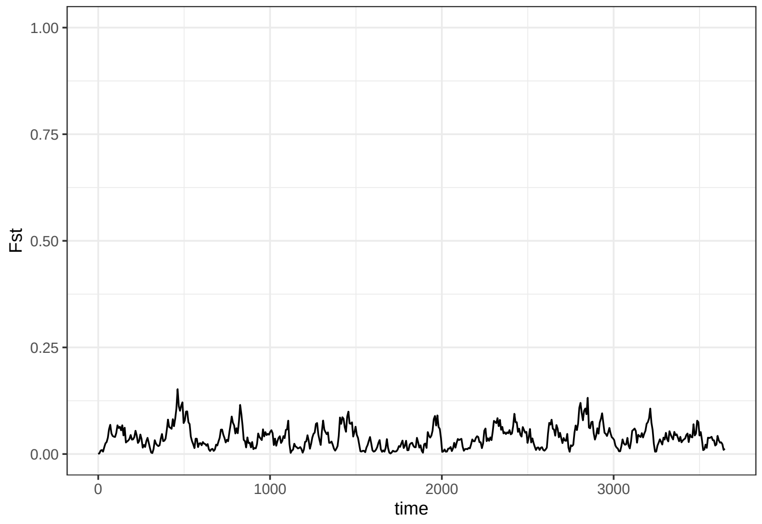
Fst with Migration Lets look at what happens to F_{ST} when we include migration.

Q13. Run the simulation above, and use the code you've seen previously to calculate and plot Fst from the results. What happens to F_{ST} when you increase/decrease the migration rate? Does this make sense?

Click For Answer

A13. With migration, allele frequencies still fluctuate but tend to move together. The higher the migration rate, the more closely correlated the results. You might also notive that fluctuations are smaller (drift is weaker) when migration is high, because our "effective" population size becomes larger.

```
# define parameters
N <- 1e3
mu <- 1e-4
m <- 1e-2

# simulate data with migration and mutation
sim1 <- sim_freqs(N = N, t_out = seq(0, 365*10, 7), demes = 5, mut_rate = mu,
                  mig_rate = m, plot_on = FALSE)

# calculate and plot Fst
sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(p_mean = mean(freq),
            p_var = popvar(freq),
            Fst = p_var / (p_mean * (1 - p_mean))) %>%
  ggplot() + theme_bw() +
  geom_line(aes(x = time, y = Fst)) +
  expand_limits(y = c(0, 1))
```



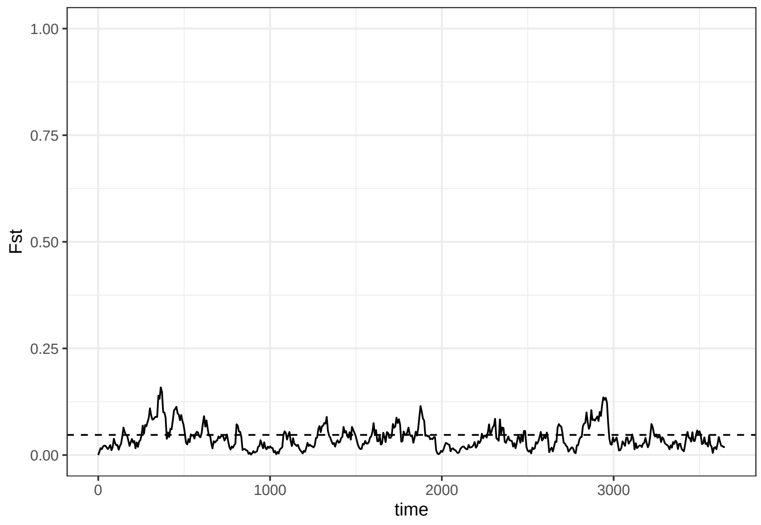
Theory tells us that the equilibrium level of F_{ST} in this situation is given by $\frac{1}{1+2N(m+\mu)}$. This is a similar idea to the equilibrium we discussed above, but we are now balancing the forces of drift, migration, and mutation to reach a new equilibrium.

Q14. Does your simulation match this equilibrium value? Note that migration rates are usually much larger than mutation rates, meaning we can often simplify the above equilibrium to $\frac{1}{1+2Nm}$.

Click For Answer

A14. The simulation matches the theory very well. When migration rates are high, Fst values are often very low.

```
# define parameters
N <- 1e3
mu <- 1e-4
m <- 1e-2

# calculate theoretical equilibrium value
Fst_eq <- 1 / (1 + 2*N*(m + mu))

# simulate data with migration and mutation
sim1 <- sim_freqs(N = N, t_out = seq(0, 365*10, 7), demes = 5, mut_rate = mu,
                  mig_rate = m, plot_on = FALSE)

# calculate and plot Fst, and overlay equilibrium
sim1$data %>%
  select(-locus) %>%
  group_by(time) %>%
  summarise(p_mean = mean(freq),
            p_var = popvar(freq),
            Fst = p_var / (p_mean * (1 - p_mean))) %>%
  ggplot() + theme_bw() +
  geom_line(aes(x = time, y = Fst)) +
  expand_limits(y = c(0, 1)) +
  geom_hline(yintercept = Fst_eq, linetype = "dashed")
```



In conclusion, at equilibrium F_{ST} reaches a level that is a balance between the strength of genetic drift and migration. Mutation plays a role, but the mutation rate is often much smaller than the migration rate meaning we can effectively ignore it. The higher the migration rate, the lower the value of F_{ST}. This means we can use F_{ST} to tell us something about the connectivity of populations. Small pairwise F_{ST} between demes tends to indicate highly connected populations and vice versa.

Note, however, that these patterns take a very long time to build up. Most of the simulations above are run over many years, and this is for a small population size - for a larger population size it may take decades to reach equilibrium. So, allele frequency changes and differentiation metrics can tell us something about connectivity on an **evolutionary timescale**. This provides important background context, but is probably not directly relevant for control purposes.

Analysis of DRC data

Now we will explore allele frequencies in a real dataset. First, we will load a dataset collected using Molecular Inversion Probes (MIPs) which are a method of high throughput sequencing. This dataset consists of 2537 samples collected in 2013–2015 from the DRC and surrounding countries. This dataset is a slightly simplified version of that used in the paper by Verity et al. 2020.

```
# load data
MIP_data <- readRDS("data/DRC_MIPs_biallelic_processed.rds")

# have a look at the data
names(MIP_data)
```

```
## [1] "coverage"       "counts"         "samples"        "loci"
## [5] "filter_history" "vcfmeta"
```

```
head(MIP_data$samples)
```

```
##                         ID Country Admin1_name Year Latitude Longitude Cluster
## 07GHR5002-AG-1 07GHR5002   Ghana        <NA> 2013 6.385908 -0.376496       1
## 07GHR5006-AG-1 07GHR5006   Ghana        <NA> 2013 6.385908 -0.376496       1
## 07GHR5018-AG-1 07GHR5018   Ghana        <NA> 2013 6.385908 -0.376496       1
## 07GHR5025-AG-1 07GHR5025   Ghana        <NA> 2013 6.385908 -0.376496       1
## 07GHR5028-AG-1 07GHR5028   Ghana        <NA> 2013 6.385908 -0.376496       1
## 07GHR5029-AG-1 07GHR5029   Ghana        <NA> 2013 6.385908 -0.376496       1
```

```
head(MIP_data$loci)
```

```
##    CHROM    POS REF ALT     NEUTRAL            GEO
## 9      1 100608   A   G     Neutral Non-geographic
## 19     1 138823   C   T     Neutral Non-geographic
## 31     1 139191   C   T Non-neutral     Geographic
## 35     1 140820   A   C Non-neutral     Geographic
## 36     1 155939   G   A     Neutral Non-geographic
## 63     1 182927   T   C     Neutral Non-geographic
```

MIP datasets like this can be analyzed using the package `MIPanalyzer`, which contains various filtering, analysis and visualisation functions. For example, we can create a copy of the dataset that filters out any DRC samples using the `MIPanalyzer::filter_samples()` function:

```
# filter samples
MIP_data_noDRC <- MIPanalyzer::filter_samples(MIP_data, MIP_data$samples$Country != "DRC")

MIP_data_noDRC
```

PCA of DRC data

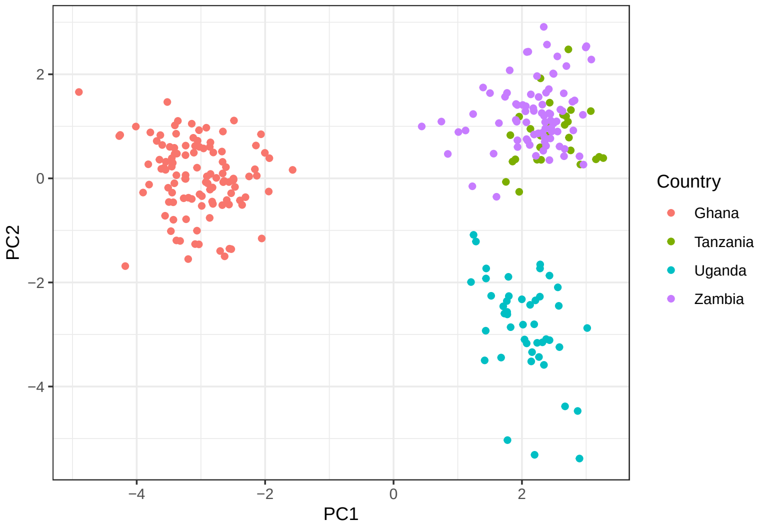
Principal Component Analysis, or PCA, can be used to reduce the number of dimensions or variables in a dataset. As such, it is often used as a way of visualising high-dimensional data, such as the allele frequencies in this example.

First, we can use the function `get_wsaf()` to calculate within-sample allele frequencies at every locus and for every individual. Then, we can use the function `pca_wsaf()` to compute PCA output from these frequencies:

```
# calculate within-sample allele frequencies. This function also imputes missing
# values by using the mean over all samples
wsaf_impute <- MIPanalyzer::get_wsaf(MIP_data_noDRC, impute = TRUE, FUN = mean)

# perform PCA analysis on within-sample allele frequencies
pca <- MIPanalyzer::pca_wsaf(wsaf_impute)
```

Let’s get our first two principal components into a dataframe and produce a scatterplot using `ggplot`:

```
# get PC1 and PC2 into dataframe
plot_df <- data.frame(PC1 = pca$x[,1],
                      PC2 = pca$x[,2],
                      Country = MIP_data_noDRC$samples$Country)

# produce scatterplot, coloured by country
ggplot(plot_df) + theme_bw() +
  geom_point(aes(x = PC1, y = PC2, color = Country))
```



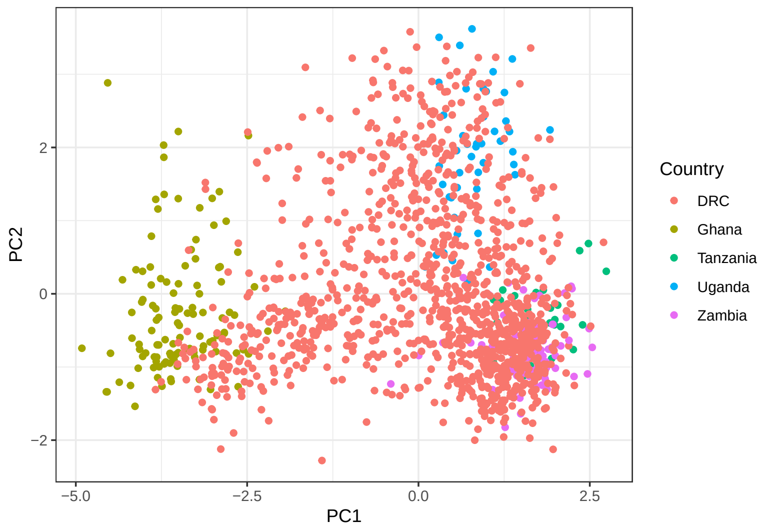
We can see several well-separated clusters, corresponding to different areas within Sub-Saharan Africa.

Q15. Repeat the process above using all samples, including those from DRC. What do you notice about the DRC samples in the scatterplot relative to the other clusters?

Click For Answer

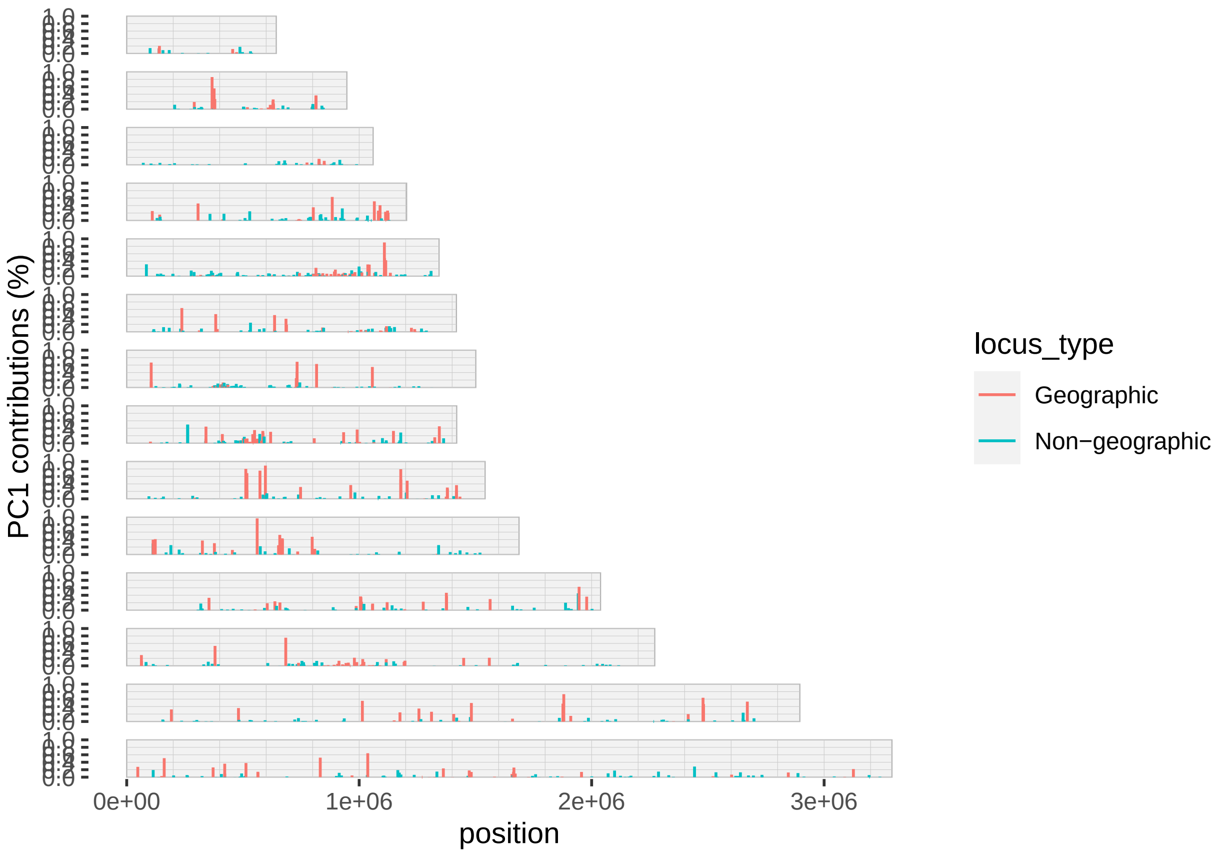
A15. The DRC samples form a large cluster that connects together the other three. Again, this makes sense geographically, because DRC lies between the other countries and so we might expect allele frequencies to be intermediate between the other clusters.

```r
# calculate within-sample allele frequencies
wsaf_impute <- MIPanalyzer::get_wsaf(MIP_data, impute = TRUE, FUN = mean)

# perform PCA analysis on within-sample allele frequencies
pca <- MIPanalyzer::pca_wsaf(wsaf_impute)

# get PC1 and PC2 into dataframe
plot_df <- data.frame(PC1 = pca$x[,1],
                      PC2 = pca$x[,2],
                      Country = MIP_data$samples$Country)

# produce scatterplot, coloured by country
ggplot(plot_df) + theme_bw() +
  geom_point(aes(x = PC1, y = PC2, color = Country))
```



Another advantage of PCA is that we can look at the contribution that each locus made to each component. These values, called *loading values*, can be used to figure out which loci are driving the observed pattern. Fortunately, the `MIPanalyzer` function `plot_pca_contribution()` takes care of this for us:

```
# plot component 1 loading values
MIPanalyzer::plot_pca_contribution(pca, component = 1, chrom = MIP_data$loci$CHROM, pos = MIP_data$loci$
                                   locus_type = MIP_data$loci$GEO)
```



Q16. What do you notice about the loading values of "Geographic" vs. "Non-geographic" loci? What does this tell you about what is driving the pattern in PC1?

Click For Answer

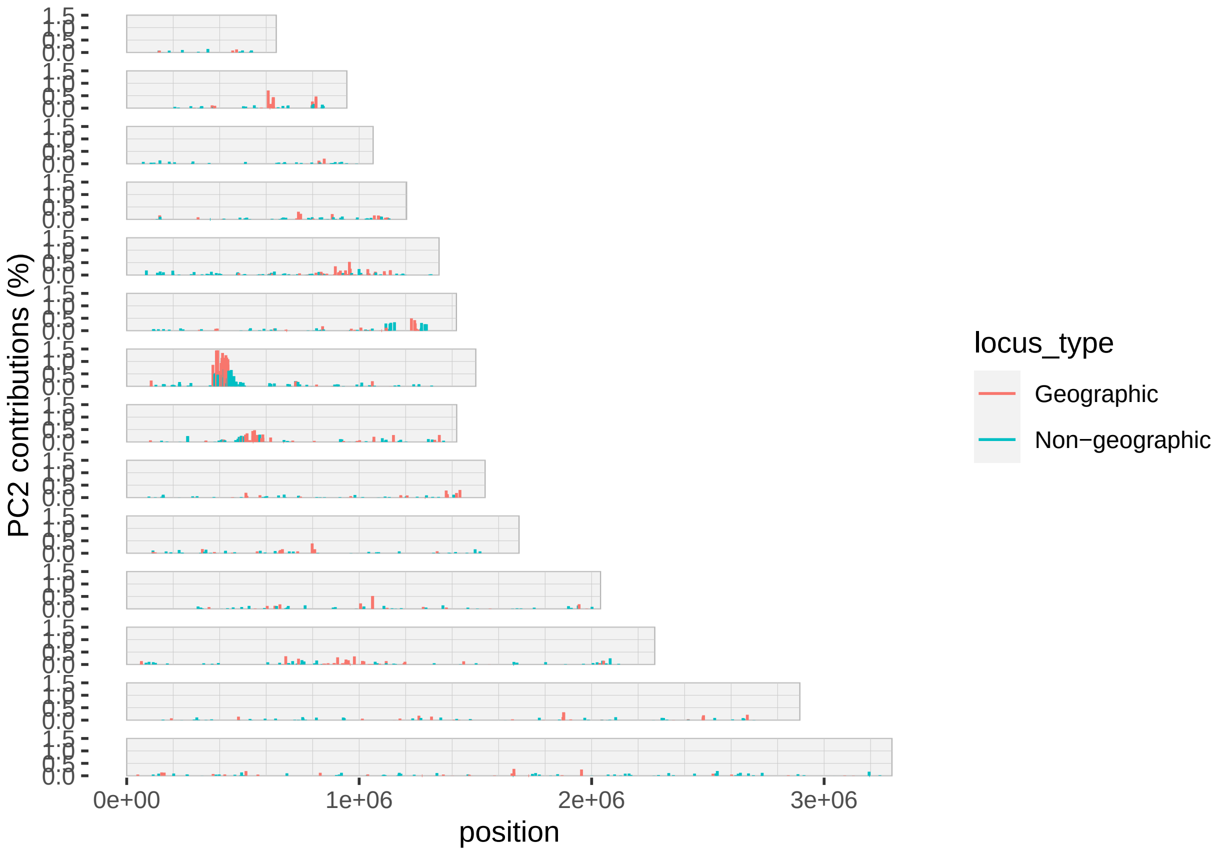
A16. The loading values in “Geographic” loci are higher on average than in the “Non-geographic” loci. These geographic loci were chosen based on having high spatial differentiation over the African continent. Therefore, this suggests that PC1 is primarily driven by space. In other words, the primary factor explaining the variation in allele frequencies that we observe in the data is what location they were sampled from.

Q17. Repeat the process above, but now looking at PC2. What do you notice about where these loci are on the genome? What genes are at these positions? You can use PlasmoDB to try and investigate these genomic locations. What does this tell you about what is driving the pattern in PC2?

Click For Answer

A17. Unlike PC1, we now see strong peaks in just a few places. These are the *pfcrt* gene on chrom7 and then a smaller peak at *dhps* on chromosome 8. This tells us that the second axis of variation in allele frequencies is primarily driven by drug resistance.

```
# plot component 2 loading values
MIPanalyzer::plot_pca_contribution(pca, component = 2, chrom = MIP_data$loci$CHROM, pos = MIP_data$loci$
                                   locus_type = MIP_data$loci$GEO)
```



In summary, PCA can be a powerful exploratory tool for finding patterns in high-dimensional datasets. When applied to allele frequencies it can be used to identify clusters (a form of population structure), and loading values can tell us which loci contribute most strongly to this pattern.

Inferring population structure using *rmaverick*

rmaverick is an R package that uses a model-based approach to investigate population structure. It is considerably more advanced than the simple PCA approach above, which has some advantages and some disadvantages. This section of the practical is therefore optional, and is really for those who want to get a look at some more advanced modelling methods. Please read the package documentation if you want to get a better understanding the various input parameters and model assumptions.

First, we will load some 24 SNP genetic barcode data from Bei et al., 2018. This study contains samples taken at two time points more than ten years apart in two locations in Senegal. We use a processed version of the raw data that has already been filtered to remove certain samples, and ordered in terms of date (2001-2002 vs. 2014) and location (Dielmo and Ndiop).

The data includes measures of transmission intensity as well as the raw barcode data. It has three elements: - Entomological Innoculation Rate (EIR) - The barcodes - The SNP locations

```
# load the processed data
load("data/Bei_2018_processed.RData")

names(Bei_2018_processed)
```

```
## [1] "EIR"           "barcodes"      "SNP_locations"
```

```
# take a peek at the various elements
head(Bei_2018_processed$EIR)
```

```
##   Time range Location   EIR
## 1       2001   Dielmo 353.8
## 2       2002   Dielmo 409.9
## 3       2001    Ndiop 171.9
## 4       2002    Ndiop  16.9
## 5       2014   Dielmo  26.3
## 6       2014    Ndiop <0.05
```

```
head(Bei_2018_processed$barcodes)
```

```
##   Sample Code       Date Sex  Age Parasite Density Location M/P genomic A1 B1
## 1        IP09 2001-01-02   F  3.4             9150   Dielmo           M  2  1
## 2        IP07 2001-01-03   F  2.6            20800   Dielmo           P  3  1
## 3        IP03 2001-02-05   M 10.5             6450   Dielmo           P NA  1
## 4        IP05 2001-02-05   F  3.8            22200   Dielmo           P  2  1
## 5        IP01 2001-02-06   M  2.3            10550   Dielmo           P NA NA
## 6        IP08 2002-01-07   F  3.7            29450   Dielmo           M  3  1
##   A2 B2 A3 B3 A4 B4 A5 B5 A6 B6 A7 B7 A8 B8 A9 B9 A10 B10 A11 B11 A12 B12
## 1  2  2  4  2  1  4  3  3  3  1  3  2  1  1  2  3   2   4   1   3   3   4
## 2  2  3  2  2  4  4  3 NA  3  1  2  2  2  2  2  1   2   1   2  NA   3   4
## 3 NA  3  2 NA NA  4 NA  3  3  1  2 NA  2 NA NA NA   1  NA   1   3   3   4
## 4  3  2  4  2  1  4  3  2  3 NA  3  1  2  2  2 NA   1   1   1  NA   3   4
## 5  2 NA  4  2 NA  4 NA  2  3  1 NA NA  2  2 NA NA   1  NA   1  NA   3   4
## 6  2  2  4  2  4  1  1  3  3  1  3  1  2  2  2  3   1   4   1   2   3   4
```

```
head(Bei_2018_processed$SNP_locations)
```

```
##   SNP code    SNP location
## 1       A1 Pf_01_000130573
## 2       B1 Pf_01_000539044
## 3       A2 Pf_02_000842803
## 4       B2 Pf_04_000282592
## 5        A3 Pf_05_000931601
## 6        B3 Pf_06_000145472
```

Unfortunately, `rmaverick` only works on monoclonal samples. Therefore let's filter out any samples identified as likely to be polygenomic:

```r
# filter out polygenomics
mav_data <- Bei_2018_processed$barcodes %>%
  filter(`M/P genomic` == "M")
```

Next, we need to load the data into *rmaverick* through the `bind_data()` function. We can also set up our first model using the `new_set()` function - this defines the assumptions of the model that we will be fitting. We can then print the project to check everything looks as planned (25 samples, 24 loci etc.)

```r
# create project, bind data and setup first model
myproj <- rmaverick::mavproject() %>%
  rmaverick::bind_data(df = mav_data, ID_col = 1, data_cols = 8:31, ploidy = 1) %>%
  rmaverick::new_set(name = "no admixture model", admix_on = FALSE)

myproj
```

```
## DATA:
##    individuals = 25
##    loci = 24
##    ploidy = 1
##    pops = 1
##    missing data = 2 of 600 gene copies (0%)
##
## PARAMETER SETS:
##  * SET1: no admixture model
##
## ACTIVE SET: SET1
##    model = no-admixture
##    lambda = 1
```

Now we are ready to run the main analysis. This uses Markov Chain Monte Carlo (MCMC) to sample from the posterior distribution of group membership. In other words, it tries to group samples together that have similar alleles over multiple loci. One important question is - how many groups (subpopulations) are there? The number of groups is given the name K in this model, and the analysis involves exploring several different values of K and working out which one we think is best supported by the data.

The following function runs the main `rmaverick` analysis. You should run this without the `quite()` wrapper so that you can see the full MCMC in action. Note that we are exploring values of K from 1 to 8 here:

```r
# run main analysis
myproj <- quiet(rmaverick::run_mcmc(myproj, K = 1:8, burnin = 1e3, samples = 1e3,
                                   rungs = 10, GTI_pow = 1.5))
```

There are various diagnostics that we should use to check that our MCMC has run as we expected. In the interest of time we will skip over these here, but please read the package documentation when running on your own data to ensure you get meaningful results.

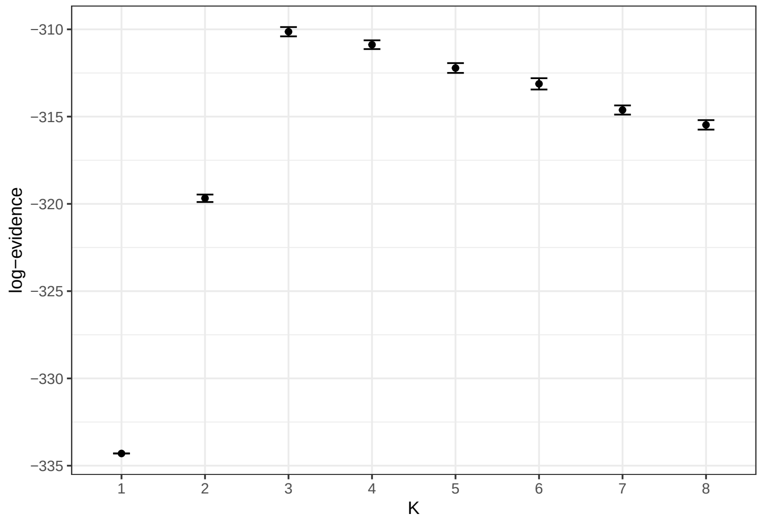
The function `plot_logevidence_K()` shows us the log-likelihood of each value of K explored. Positive values (or less negative values) provide greater support for K. The function `plot_posterior_K()` works in a very similar way, but can be interpreted as an ordinary probability.

Q18. Use the `plot_logevidence_K()` and`plot_posterior_K()` functions to explore which value of K are best supported in this case. Which values are supported?

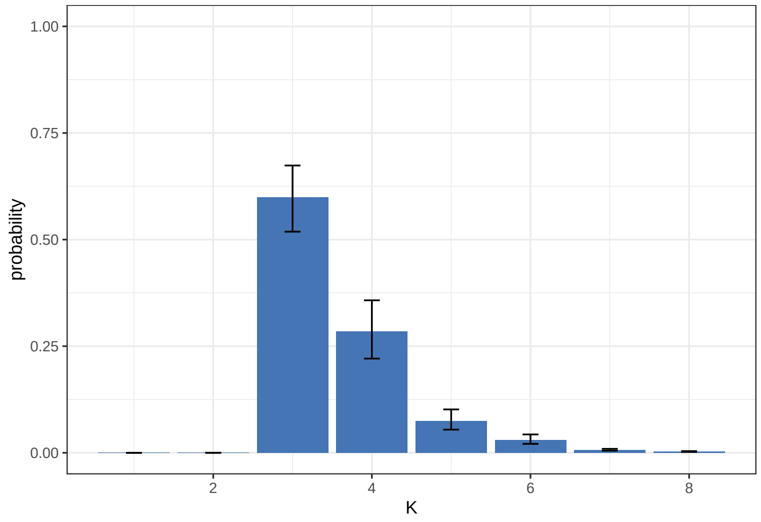
Click For Answer

A18. From both plots it is clear that $K = 3$ is best supported in this case. From the probability plot we can see that values of $K = 4$ and $K = 5$ are also plausible.

```
# plot in log space
rmaverick::plot_logevidence_K(myproj)
```



```
# plot as probability
rmaverick::plot_posterior_K(myproj)
```

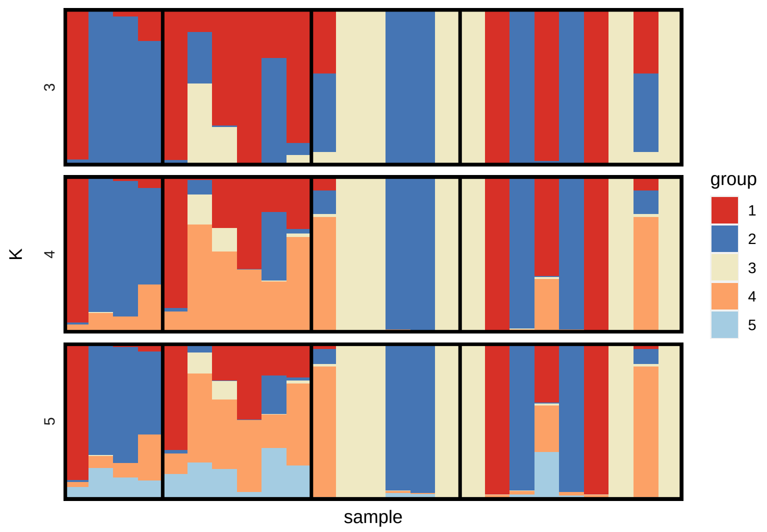


Finally, we can use the `plot_qmatrix()` function to produce a "STRUCTURE" plot, named after the original STRUCTURE program on which `rmaverick` is based. This plot can be interpreted in the following way:

- Each stacked bar represents a sample.
- The proportion of each colour represents the probability of that sample belonging to each cluster. For example, if a bar is 50% red and 50% blue then this sample has an equal chance of belonging to each of these two clusters.
- We often produce several STRUCTURE plots at once for different values of K. The number of colours present in the plot will equal K.

In this case we also add vertical lines to break up the genetic data into the following groups: 1. Dielmo 2001-2001 2. Ndiop 2001-2002 3. Dielmo 2014 4. Ndiop 2014

```r
rmaverick::plot_qmatrix(myproj, K = 3:5) +
  geom_vline(xintercept = c(4, 10, 16) + 0.5, size = 1)
```



Notice that the later samples are (mostly) clearly allocated to one or other population with high probability. These essentially represent clonal lineages. We can see that groups 2,3,4 are present in Dielmo in 2014, and in Ndiop we additionally have group 1.

Comparing this to the earlier time point, samples are far more ambiguous in their allocation. They look a bit like some of these clonal lineages, but it's not clear cut. Interestingly, group 4 dominates in Ndiop in 2001-2002, despite being almost completely absent from Ndiop in 2014.

Q19. Overall, we can see evidence of increasing population structure over time. Now look at the EIR values, also stored within the same dataset. Can you make sense of what is likely happening here?

Click For Answer

A19. There is lower transmission at the later timepoint, therefore it is not surprising that the population is becoming more clonal. We also see increasing differentiation over time, as we would expect if the populations were partially isolated. However, we should keep in mind that this is just a small sample at each timepoint, and therefore it is likely that many genotypes in the population were missed.